AI Shopping and Recipe Assistant Chatbot Documentation

Overview

The AI Shopping and Recipe Assistant is a React Native chatbot that combines product search capabilities with recipe generation. It's designed to help users find products and generate recipes using natural language processing. The chatbot provides a user-friendly interface with real-time responses and multiple AI model integrations.

Core Features

- Product availability checking
- Recipe generation from ingredients
- FAQ handling
- Natural language conversation
- Real-time response simulation
- Interactive UI with typing indicators

Setup Instructions

1. API Keys

The chatbot uses Hugging Face's API for general conversation and recipe generation. You'll need to:

1. Create an account on Hugging Face (https://huggingface.co)
2. Generate an API key from your account settings
3. Replace the API key in the code:

```
const API_KEY = 'YOUR_HUGGING_FACE_API_KEY';  // Replace this
```

2. Backend Setup

The chatbot requires a product database backend. Current configuration:

```
const response = await axios.get('http://localhost:5002/products');
```

You'll need to:

1. Set up your product database
2. Update the endpoint URL to match your backend
3. Ensure your product data follows the Product interface structure

```
interface Product {
  _id: string;
  current_price: number;
  link_image: string;
  product_code: string;
  product_id: string;
  product_name: string;
  category: string;
  link: string;
  measurement: string;
  sub_category_1: string;
  sub_category_2: string;
  unit_per_prod: string;
}
```

Key Components

1. ConversationContext Class

This class manages the conversation history and context:

- Maintains a history of up to 10 messages
- Tracks the last query
- Provides methods for context management

2. Message Handler System

The **handleAIResponse** function processes messages through multiple stages:

1. FAQ checking
2. Recipe query detection
3. Product availability checking
4. General conversation fallback

Quick Response System

The chatbot includes a pre-defined FAQ system that can be easily modified:

```
const quickResponses = {
  "How does DiscountMate work?": "DiscountMate compares prices...",
  "Where do you get your price data?": "We collect price data...",
  // Add more FAQs here
```

```
};
```

To add new FAQs, simply extend this object with new question-answer pairs.

Changing AI Models in the Chatbot

Model Configuration

1. General Conversation Model:

Located in the handleAIResponse function:

```
const response = await axios.post(
  'https://api-inference.huggingface.co/models/facebook/blenderbot-400M-distill',//Replace the URL with your chosen model's endpoint
  {
    inputs: message,
    parameters: {
      max_length: 100,
      temperature: 0.7,
      top_p: 0.9,
    }
  },
  {
    headers: {
      'Authorization': `Bearer ${API_KEY}`,
      'Content-Type': 'application/json',
    },
  }
);
```

To change the conversation model:

1. **Replace the URL with your chosen model's endpoint**
2. **Adjust the parameters based on the new model's requirements**
3. **Example for GPT-2:**

```
4.    const response = await axios.post(
5.      'https://api-inference.huggingface.co/models/gpt2',
```

```
    {
      inputs: message,
      parameters: {
        max_length: 150,  // Adjust based on needs
        temperature: 0.8,
        return_full_text: false,
      }
    },
    {
      headers: {
        'Authorization': `Bearer ${API_KEY}`,
        'Content-Type': 'application/json',
      },
    }
  );
```

2. Recipe Generation Model

Located in the generateRecipe function:

```
const response = await axios.post(
  'https://api-inference.huggingface.co/models/flax-community/t5-recipe-generation', ,//Replace the URL with your chosen model's endpoint
  {
    inputs: `ingredients: ${ingredients}`,
    parameters: {
      max_length: 512,
      temperature: 0.7,
      top_p: 0.95,
      do_sample: true
    }
  },
  {
    headers: {
      'Authorization': `Bearer ${API_KEY}`,
      'Content-Type': 'application/json',
    },
  }
);
```

To change the recipe model:

1. **Replace the URL with your chosen model's endpoint**
2. **Adjust the input format based on the new model's requirements**
3. **Example for a different recipe model:**

```
const response = await axios.post(
  'https://api-inference.huggingface.co/models/your-chosen-model',
  {
    inputs: {
      ingredients: ingredients,
      style: "detailed",  // Add additional parameters as needed
    },
    parameters: {
      max_length: 1000,
      temperature: 0.8,
      num_return_sequences: 1,
    }
  },
  {
    headers: {
      'Authorization': `Bearer ${API_KEY}`,
      'Content-Type': 'application/json',
    },
  }
);
```

Important Considerations When Changing Models

1. Response Format Handling

Different models return different response formats. You'll need to adjust the response handling:

```
let recipe = response.data[0].generated_text;

// May need to change to something like:
let recipe = response.data.choices[0].text;  // For OpenAI-style responses
// or
let recipe = response.data.output[0];  // For other API formats
```

2. Error Handling

Add appropriate error handling for your chosen model:

```
try {
  const response = await axios.post(/* ... */);

  // Add model-specific error checking
  if (!response.data || response.data.error) {
    throw new Error('Model returned invalid response');
  }

  // Process response
} catch (error) {
  if (error.response?.status === 429) {
    // Handle rate limiting
    return "I'm getting too many requests. Please try again in a moment.";
  }
  // Handle other errors
  console.error('Model Error:', error);
  return "I'm having trouble processing your request.";
}
```

3. Model-Specific Parameters

Each model may have different parameter requirements:

- **temperature: Controls randomness (0.0 to 1.0)**
- **max_length: Maximum token length for response**
- **top_p/top_k: Sampling parameters**
- **Model-specific parameters (needs to be documented for each model)**

AI Model Analysis and Comparison of the Models I Tested

Shopping Assistant Models Tested

1. BlenderBot (facebook/blenderbot-400M-distill)

Endpoint: https://api-inference.huggingface.co/models/facebook/blenderbot-400M-distill

Strengths:

- Excellent conversational abilities

- Good context retention
- Fast response times (avg. 1-2 seconds)
- Handles multiple topics within same conversation
- Memory efficient (400M parameters)

Drawbacks:

- Limited product-specific knowledge
- Can sometimes generate overly casual responses
- Occasional repetition in longer conversations

Best For:

- General customer service interactions
- Basic product inquiries
- Multi-turn conversations

2. FLAN-T5 (google/flan-t5-large)

Endpoint: https://api-inference.huggingface.co/models/google/flan-t5-large

Strengths:

- Excellent at structured responses
- Strong reasoning capabilities
- Good at extracting product information
- Consistent output format

Drawbacks:

- Slower response times (avg. 2-3 seconds)
- Less conversational than BlenderBot
- Higher resource usage
- Can be overly formal

Best For:

- Detailed product specifications
- Complex query understanding
- Information extraction tasks

3. GPT-2 (gpt2)

Endpoint: https://api-inference.huggingface.co/models/gpt2

Strengths:

- More efficient than larger models
- Good balance of speed and accuracy
- Decent understanding of product context
- Flexible response generation

Drawbacks:

- **Less sophisticated than newer models**
- **Can sometimes generate generic responses**
- **Limited context window**
- **May need more prompt engineering**

Best For:

- **Quick product descriptions**
- **Basic customer support**
- **General queries**
- **High-traffic scenarios**

Recipe Generation Models Tested

1. T5 Recipe Generator (flax-community/t5-recipe-generation)

Endpoint: https://api-inference.huggingface.co/models/flax-community/t5-recipe-generation

Strengths:

- Specifically trained for recipes
- Consistent format
- Good ingredient proportions
- Clear step-by-step instructions

Drawbacks:

- Limited creativity
- Can be repetitive
- Fixed output structure
- Sometimes generates common recipes regardless of input

2. DUT Recipe Generator (Ashikan/dut-recipe-generator)

Endpoint: https://api-inference.huggingface.co/models/Ashikan/dut-recipe-generator

Strengths:

- Specifically trained for recipe generation
- Better understanding of ingredient combinations
- More creative recipe suggestions
- Includes cooking tips and techniques
- Good at generating diverse recipes

Drawbacks:

- May require more specific ingredient details
- Response time can be slightly longer (2-3 seconds)
- Output might need formatting cleanup
- Limited to certain cuisine types

Comprehensive Enhancement Recommendations for AI Cooking Assistant

1. Smart Recipe & Dietary Management

- Generate recipes based on dietary restrictions, skill level, and preferences
- Include nutritional information, ingredient substitutions, and portion scaling
- Track calories, allergens, macro/micronutrients, and health goals
- Provide dietary compliance checking and meal planning for health conditions
- Integrate with fitness apps and save favourite recipes
- Create meal prep schedules and weekly meal plans
- Rate recipes and track nutritional goals

2. Intelligent Shopping & Inventory System

- Track real-time product availability and compare prices across stores
- Generate smart shopping lists based on recipes and missing ingredients
- Calculate total recipe costs and suggest budget alternatives